Validation d'applications temps réel distribuées autour du réseau CAN

Nicolas NAVET, LORIA, Equipe TRIO
Ensem, 2 Av. de La forêt de Haye – 54516 Vandoeuvre
nnavet@loria.fr - http://www.loria.fr/~nnavet

Ce papier se propose de donner une synthèse des techniques de validation d'applications temps réel distribuées autour d'un réseau CAN, la validation étant la vérification du bon respect des contraintes de l'application qui seront pour nous essentiellement des contraintes de temps et de sûreté de fonctionnement. Il existe deux grandes techniques de validation, l'exécution de modèles, qu'ils s'agissent de modèles analytiques ou de modèles de simulation, et l'observation sur prototypes ou maquettes. Après avoir discuté des avantages et inconvénients de chacun, nous verrons concrètement quels résultats en attendre dans le cadre de CAN.

I] L'étape de validation dans le cycle de vie de l'application

Les contraintes de temps et de sûreté de fonctionnement (SdF) sont présentes sous une forme plus ou moins formelle dès la phase d'analyse des besoins, c'est à dire lors de l'élaboration du cahier des charges. Par exemple, si l'on considère un système de contrôle aérien, une contrainte plausible est que "la position des avions sur l'écran radar devra être mise à jour toutes les 100ms".

La vérification des propriétés temporelles commence effectivement lors de la phase de spécification dont l'objectif est la définition d'une architecture fonctionnelle validée, c'est à dire sans ambiguïtés, sans incohérences et complète [Simonot-Lion et al., 1995]. L'Architecture Fonctionnelle (ou AF) décrit formellement la structure d'une application et son comportement attendu, c'est à dire l'ensemble des services et traitements respectivement à fournir et effectuer par les fonctions ainsi que les échanges de données inter-fonctions. L'AF ne tient compte ni de la répartition des fonctions sur les sites ni de l'Architecture Matérielle (AM) support définie comme l'ensemble des composants qui assurent les ressources matérielles et logicielles nécessaires à l'exécution de l'application (e.g. machines, OS, réseaux, protocoles, ...). L'indépendance de l'AF vis-à-vis du support exécutif permet d'une part de réduire la complexité du problème général et d'autre part de faciliter la réutilisation d'AFs existantes. Néanmoins, dans cette étape du cycle de vie, une validation ne peut être menée qu'en faisant des hypothèses fortes sur les durées de traitement et donc implicitement sur les performances. Les vérifications sur l'AF portent dont essentiellement sur la correction "logique" du système, comme par exemple, l'absence d'interblocage, la terminaison des traitements ou l'équité dans l'accès à une ressource.

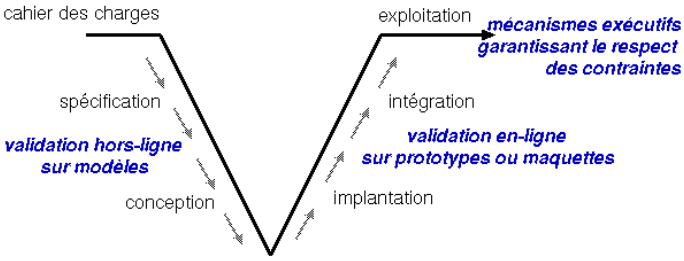

Figure 1 : La validation dans le cycle de vie de l'application

Une Architecture Opérationnelle (AO) validée est l' objectif attendu de la phase de conception du système. L' AO est le résultat de la projection (mapping) de l' AF sur l' AM support, les composants de cette dernière étant choisis relativement à un ensemble de critères issus du cahier des charges parmi lesquels les performances évidemment, le coût, qui pour les productions de masse revêt une importance toute particulière, mais aussi par exemple, des garanties sur la pérennité de la production. La projection comprend le placement des fonctions sur les différents sites mais aussi la fixation des paramètres des mécanismes exécutifs associés à l' implantation, comme les ordonnanceurs de tâches (scheduler) et les protocoles de communication. Une fois réparties sur les différents sites, les fonctions pourront être subdivisées en différentes tâches auxquelles il faudra vraisemblablement allouer une priorité et éventuellement une politique d' ordonnancement si l' OS support permet un choix. De façon similaire, selon le protocole de transmission de données, il peut être nécessaire de fixer la priorité des trames échangées, priorités qui contrairement à celles des tâches, seront globales sur l' ensemble du système sous l' hypothèse d' un médium de communication partagé. Il s' agit ensuite de s' assurer que l' AO proposée vérifie les contraintes de l' application, c' est la phase cruciale de validation de l' AO. Il est évident que la remise en cause éventuelle de choix effectués en termes d' AM, de répartition ou de fixation de paramètres sera plus facile

et coûteuse lors de l' étape de conception que plus en aval dans le cycle de vie de l' application et particulièrement pendant l' exploitation du système.
Une fois la conception terminée commence la phase d' implantation qui consiste d' une part à programmer les fonctions ou tâches et d' autre part à câbler physiquement les connexions entre sous-systèmes. Vient ensuite la phase d' intégration sur site puis naturellement l' exploitation du système assortie épisodiquement d' activités de maintenance.

II] les techniques de validation

Nous distinguons deux grandes classes de techniques de validation :

- la validation sur modèles,
- la validation sur prototype ou maquette.

Il existe naturellement des techniques de validation hybrides dans lesquelles une partie du système est simulée et l'autre implémentée sous forme d'une maquette.

Les formalismes de modélisation sont extrêmement nombreux mais en règle générale on peut dire que plus leur pouvoir d' expressivité est grand, plus leur exploitation sera problématique; il suffit pour s' en convaincre de penser au langage naturel. Nous distinguerons les modèles de simulation, c' est à dire les modèles qui se "résolvent" par simulation, et les modèles analytiques dont la résolution est purement calculatoire.

II.1] La simulation est souvent incontournable

L' utilisation de la simulation dans l' objectif d' évaluer les performances d' une AO, et donc sa capacité à respecter les contraintes de l' application, est une pratique extrêmement répandue qui s' impose généralement faute d' alternatives :

- des mesures sur un système réel nécessite sa réalisation et son instrumentation ce qui est long et coûteux et qui ne peut intervenir que loin en avant dans la phase de conception,
- les modèles analytiques ne sont, en règle générale, pas capable de refléter toute la complexité d' une AO ni de donner des renseignements sur toutes les métriques de performances imaginables: ils ne peuvent modéliser que des systèmes relativement peu complexes ou des sous-ensembles de systèmes complexes et sont donc généralement intrinsèquement insuffisants.

II.2] Limites des approches par simulation

Si la simulation est d' un emploi plus aisé que la collecte de mesures sur un système existant ou la modélisation analytique, nous avons identifié un certain nombre de difficultés qui nous ont confortés dans l' idée que généralement la simulation seule n' est pas, pour les applications temps réel, une technique de vérification suffisante, et ce à plusieurs titres :

- il pèse toujours une incertitude forte sur la correction de l' implantation d' un modèle et sur sa pertinence et donc sur la validité même des résultats de simulation.
- la simulation d' une AO fait nécessairement des hypothèses simplificatrices sur le système physique contrôlé. Ainsi, le modèle de l' AO du système de contrôle-commande embarqué dans un véhicule intègre un modèle, généralement simpliste, du comportement des différents organes du véhicule.
- les résultats dans le pire des cas (e.g. temps de réponse maximum) n' ont qu' une valeur indicative puisque simuler consiste à dérouler un certain nombre de trajectoires qui dans le cas général, est infiniment inférieur au nombre de trajectoires possibles.

Dans une optique temps réel et SdF, le dernier argument en défaveur de la simulation est le plus fort puisque, avant de penser aux performances dans le cas probable, c' est d' abord le pire cas, même si il ne survient qu' exceptionnellement, qui nous intéresse. Il nous est donc apparu naturel de "coupler" les modèles de simulation avec des modèles analytiques donnant des bornes sur les métriques de performance considérées. Cette approche a été implantée pour la première fois au sein de notre équipe dans l' atelier de construction et de validation des systèmes embarqués dans l' automobile VACANS (VAlidation of CAN based Systems) qui a fait l' objet d' une diffusion commerciale par la société DELTA PARTNERS (ancien distributeur d'OPNET) et a été utilisé dans un contrat industriel [Navet and Song, 1996] portant sur la validation d' une application embarquée dans un véhicule prototype de la société PSA (Peugeot-Citroën Automobiles). Initialement [Bélissent, 1996], cet outil se constituait d' un modèle de simulation de profils de communication CAN couplé à un calcul analytique de bornes sur le temps de latence des messages développé à l' Université de York [Tindell and Burns, 1994; Tindell et al., 1995]. Nous l' avons enrichi en développant deux modèles analytiques visant à évaluer la "robustesse" d' une application dans un environnement soumis à des perturbations électromagnétiques pouvant engendrer des erreurs de transmissions. Le premier modèle permet le calcul de la probabilité de non-respect des échéances des messages dans le pire cas [Navet et al., 2000], le second, le temps moyen d' atteinte de l' état "bus-off" [Navet and Song, 2001] et de l'état "erreur-passive" [Gaujal and Navet, 2001]. Ces deux informations n' auraient pu être obtenues par simulation; la première est une borne supérieure donc non évaluable par simulation, l' atteinte des états bus-off et erreur-passive, sous des hypothèses d' erreurs réalistes, sont de évènements trop rares pour que des simulations de durées raisonnables donnent des résultats statistiquement valables.
Les résultats que l' on peut obtenir en résolvant des modèles, qu' ils soient analytiques ou de simulation, ne sont valables qu' à la condition que ses modèles soient pertinents et que les hypothèses faites sur le système soient respectées. Par l' observation d' un prototype ou d' une maquette du système, il est possible, dans une

certaine mesure, de valider les modèles, de fixer les valeurs de certains paramètres et surtout de vérifier que certaines hypothèses sont respectées. Ainsi, dans le cadre des applications embarquées dans les véhicules, un observateur réseau est indispensable ne serait-ce que pour s' assurer que les différents organes électroniques du véhicule, qui sont majoritairement conçus et fabriqués par des équipementiers, respectent leur spécification en termes de trafic réseau généré. D' autre part, les modèles sont généralement très grossiers quant à la partie opérative du système et le respect de certaines contraintes, que nous qualifierons de contraintes de niveau applicatif et dont l' exemple typique est la contrainte de temps entre un changement de vitesse et la réduction de couple associée, n' est pas vérifiable. Toutes ses raisons nous ont conduits à développer dans le cadre d' un second contrat industriel avec PSA [Song and Navet, 1997], un observateur réseau "intelligent" capable de vérifier en-ligne le respect de certaines contraintes de niveau applicatif que l' on aura spécifiées au préalable.

Le tableau 1 présente avantages et inconvénients de chacune des techniques de validation.

Validation sur Modèles		**Validation sur Prototypes**
+ coûts et rapidité de développement - Hypothèses simplificatrices		+ pas d'hypothèse simplificatrice + certaines contraintes ne peuvent être vérifiées que sur prototype ! - construction, instrumentation longues et coûteuses - intervient tard dans la phase de conception - résultats orientés pire-cas généralement sans validité
Modèles Analytiques	**Modèles de Simulation**	
+ validité du modèle vérifiable + bien adaptés aux contraintes dans le pire-cas - hypothèses simplificatrices fortes - difficultés de modélisation !	+ moins d'hypothèses simplificatrices que les modèles analytiques + facilité de modélisation (existence d'outils) + toutes les métriques de performances sont possibles - résultats orientés pire-cas généralement sans validité - validité des modèles !	

Tableau 1 : Avantages et inconvénients des différentes techniques de validation.

Dans la suite du document, nous ne développerons pas les travaux portant sur la construction de modèles de simulation et renvoyons le lecteur à [Bélissent, 1996], [Kiencke et al., 1997], [Courrier et al., 1998] et [Castelpietra et al. 2001] pour des travaux très récents.

III] Modèles Analytiques : Quels résultats ?

Dans cette section, nous examinerons différentes métriques de performances qui peuvent être évaluées de façon réaliste à l'aide de modèles analytiques. Sans rentrer dans tous les détails mathématiques (les références appropriées seront données), l'objectif est de présenter les résultats et leur technique d'obtention.
Les analyses présentées dans cette section s' appliquent aux messages périodiques ou sporadiques, dans ce dernier cas, en considérant la période égale au temps minimum inter-arrivée. L' ensemble des messages de l' application est noté $M = \{m_1, m_2, ..., m_k\}$ ou m_k est le message de priorité *k*, de période T_k et d'échéance D_k .

III.1] Bornes sur les temps de réponse

Intérêt: les informations échangées par les différentes station d'un réseau CAN sont généralement de durée de vie limitée et il s'agit donc de garantir la fraîcheur des données consommées. Cela implique que le temps de réponse d'un message, défini comme le temps entre l' activation de la tâche qui produit m_k, notée τ_k et la réception complète du message par le ou les destinataires, doit être toujours inférieur à l'échéance du message.
Formalisme utilisé: analyse d'ordonnançabilité (calcul du point fixe d'une fonction)
Références bibliographiques: [Tindell and Burns, 1994] pour a) et b), [Tindell and Clark, 1994] pour c).

III.1.a) temps de réponse avec transmission fiable

Le message m_k hérite sa période de τ_k mais entre deux mises à disposition successives du message, il se peut qu' il y ait une certaine variabilité (appelée aussi gigue). En effet, le temps entre l' activation de la tâche τ_k et l' invocation de la requête de transfert de données de m_k dépend, par exemple, de l' activation ou non de tâches plus prioritaires sur le processeur du noeud considéré. On note J_k la borne sur la gigue du message m_k. Si l' on n' a pas d' indication sur l' instant auquel τ_k génère m_k au cours de son exécution, par sécurité on peut considérer que m_k n' est disponible qu' à la toute fin de l' exécution de τ_k. J_k est alors égal au pire temps de réponse de τ_k.
La durée de transmission C_k d' un message m_k pouvant être bornée, pour calculer le pire temps de réponse il nous suffit de calculer le temps maximal que le message m_k devra attendre avant de gagner l' accès au bus (ce temps noté I_k est aussi appelé le temps d' interférence). L' accès au bus peut être bloqué par:

- des messages plus prioritaires, qui peuvent, selon leurs périodes, être transmis une ou plusieurs fois pendant I_k ,

- un message moins prioritaire en cours de transmission, et dans le pire des cas, ce temps sera la durée de transmission de la plus longue trame moins prioritaire.

le temps de réponse du message m_k est donc

$$R_k = C_k + J_k + I_k \quad (1)$$

avec I_k la limite, quand n tend vers l' infini, de la suite

$$I_k^0 = 0, I_k^n = \max_{i>k}(C_i) + \sum_{j<k} \left\lceil \frac{I_k^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil \quad (2)$$

I_k est calculé jusqu' à convergence ou jusqu' à ce que $I_k^n > D_k - C_k$. Dans ce dernier cas, il n' y a pas de garantie que m_k respecte son échéance et M est dit non-ordonnançable.

III.1.b) temps de réponse avec erreurs de transmission

Cette analyse peut être effectuée en considérant la possibilité d' occurrences d'un nombre fini d'erreurs de transmission. Pour cela, [Tindell and Burns, 1994] proposent le modèle d' erreurs suivant :

- quelque soit le temps t considéré, on a au plus une rafale d' erreurs de n_{error},
- mise à part cette rafale, deux erreurs consécutives sont espacées au minimum d' un temps T_{error}.

Ce modèle d' erreurs, "déterministe" dans la mesure où le nombre d' erreurs pendant un temps peut être borné par $\left(n_{error} + \left\lceil \frac{t}{T_{error}} \right\rceil - 1 \right)$, permet le calcul du pire temps de réponse en remplaçant (2) par

$$I_k^0 = 0, I_k^n = E_k(I_k^{n-1} + C_k) + \max_{i>k}(C_i) + \sum_{j<k} \left\lceil \frac{I_k^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil$$

avec E_k la fonction qui prend en compte les overheads induits par les erreurs :

$$E_k = \left(n_{error} + \left\lceil \frac{t}{T_{error}} \right\rceil - 1 \right) \cdot \left(23\tau_{bit} + \max_{j \le k}(C_j) \right)$$

L' overhead induit par une erreur est composé d' une part de la trame d' erreur (23 τ_{bit} dans le pire cas considéré) et d' autre part de la retransmission de la trame corrompue. Dans le pire cas, toutes les erreurs se produisent au dernier bit de la plus longue trame susceptible d' être retransmise.

III.1.c) temps de réponse de bout-en-bout

Il est possible de vérifier les contraintes de date au plus tard portant sur une transaction, c'est à dire sur une séquence de tâches s'exécutant sur plusieurs processeurs et communiquant par envoi de messages sur le bus CAN. Imaginons par exemple dans un véhicule qu'une tâche A s'active sur l'occurrence d'un changement de vitesse. Cette tâche aura, entre autres charges, celle d'émettre à la fin de son exécution un message à destination du Contrôle-moteur (CM) pour l'informer de la situation courante. Sur le site CM, sur réception du message, la tâche B qui devra éventuellement agir sur les actionneurs se déclenche. La séquence changement de vitesse – déclenchement de A – transmission d'un message sur le bus – activation de B – consignes aux actionneurs est une transaction et le délai de « bout-en-bout » pourra être borné par exemple pour des raisons liées à la mécanique.

Toutes les tâches, tous les messages de la transaction vont hériter du « pattern d'activation » de l'évènement déclencheur de la transaction. En général, il s'agira d'un évènement sporadique et la période de chaque tâche/message sera fixée comme le temps minimal entre deux arrivées de cet événement.

Pour trouver une borne sur le délai de bout-en-bout, une analyse d'ordonnançabilité est effectuée sur chaque ressource (bus, processeur) mais le temps de réponse d'une tâche (resp. d'un message) dépend de l'ordonnancement des tâches et des messages qui précédent dans la transaction. Plus précisément, la gigue sur l'activation d'une tâche (resp. d'un message) est égale au pire temps de réponse du message déclencheur (resp. de la tâche qui va émettre le message). La borne sur le délai de la transaction sera finalement le temps de réponse de la dernière tâche de la transaction. Notons que lorsque de nombreux sites sont impliqués dans une transaction, cette analyse se révèle très pessimiste du fait de l'augmentation progressive des gigues sur les dates d'activation.

III.2] Probabilité de non-respect des échéances dans le pire-cas

Intérêt : considérant qu' il est souvent irréaliste de faire l' hypothèse d' un médium 100% fiable (comme par exemple dans le cadre des applications embarquées dans des véhicules) et étant donnée que d' autre part nous ne savons pas donner une borne sur le nombre d' erreurs pouvant survenir dans un intervalle de temps (hypothèse faite au §III.1.b), il est nécessaire d' évaluer de manière probabiliste le risque qu' une trame ne respecte pas son échéance. Cette information donnera des éléments de réponse au concepteur dans le choix des priorités et du bon support physique de transmission (paire torsadée non-blindée, blindée ou fibre optique ?).

Formalisme utilisé: analyse d'ordonnançabilité, probabilités.

Références bibliographiques: [Navet and Song, 1998], [Navet et al., 2000], [Rüdiger, 1998]

Nous proposons un modèle stochastique d' erreurs, représenté sur la figure 2, qui considère à la fois la fréquence des perturbations (le nombre d' occurrences suit une loi de Poisson) et la gravité des perturbations (lorsqu' une perturbation survient, elle entraîne soit une erreur individuelle, soit une rafale d' erreurs). Dans la pratique, on observe effectivement deux types d' erreurs:des erreurs individuelles et des erreurs en rafales qui se produisent dans des zones fortement perturbées pour les véhicules ou lors de la mise en

marche ou de l' arrêt d' équipements électriques voisins du réseau.

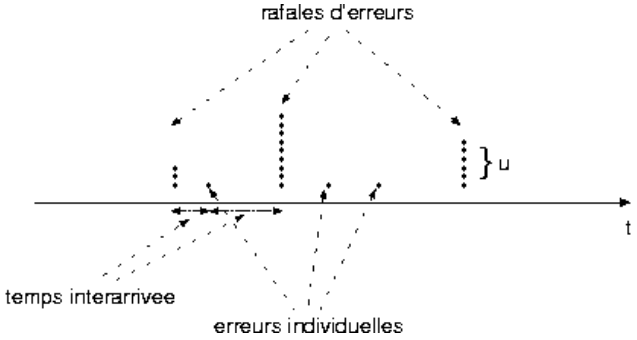

Figure 2 : le modèle d'erreurs.

On note $X(t)$ le processus stochastique qui compte le nombre d'erreurs dans le temps :
$X(t) = \sum_{i=0}^{N(t)} y_i$ avec $y_0 = 0$, où

- $N(t)$ suit une loi de Poisson de paramètre λ
- $y_i = \begin{cases} u, avec\ proba\ \alpha \\ 1, avec\ proba\ 1-\alpha \end{cases}$

avec y_i la variable aléatoire (v.a.) qui donne la taille de l' erreur (i.e. erreur individuelle ou rafale d' erreurs) et u la v.a. qui donne la taille des rafales. La distribution de u et la valeur des paramètres seront fixés à l' aide de mesures collectées sur un prototype ou avec l' expérience accumulée sur des systèmes similaires. Notons que dans cette optique, certains contrôleurs de communication CAN possèdent des fonctionnalités intéressantes comme des compteurs d' erreurs accessibles en lecture ou même a possibilité de générer une interruption sur l' occurrence d' une erreur.

Une fois les paramètres du modèle d'erreurs fixés, nous calculons pour chaque trame de l'application, le seuils d'erreurs tolérables η_k telle que la contrainte d'échéances associée au message m_k soit respectée. Si l' on note $R_k(i)$ le temps de réponse du message avec i erreurs de transmission, il s' agit de trouver pour chaque message :
$\eta_k = \max\{n \in \mathrm{N} | R_k(n) \le D_k\}$ avec

$R_k(i) = C_k + J_k + I_k(i)$ et $I_k(i)$ la limite de la suite

$I_k^0 = 0, I_k^n = \varepsilon_k(i) + \max_{i>k}(C_i) + \sum_{j<k} \left\lceil \frac{I_k^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil$ où

$\varepsilon_k(i)$ est la fonction qui intègre l'overhead induit par i erreurs de transmission :

$\varepsilon_k(i) = i.\left(23\tau_{bit} + \max_{j \le k}(C_j)\right)$

Si moins de $\eta_k + 1$ erreurs se produisent entre la mise à disposition de la trame m_k et sa fin de transmission, c' est à dire pendant le temps $R_k(\eta_k)$, alors la trame m_k respectera forcément son échéance. La probabilité de non-respect de l' échéance dans le pire cas (worst case deadline failure probability) est définie comme la probabilité qu' il y ait plus de η_k erreurs pendant le temps $R_k(\eta_k)$:

$P[X(R_k(\eta_k)) > \eta_k] = 1 - \sum_{i=0}^{N(t)} P[X(R_k(\eta_k)) = i]$

Cette probabilité est qualifiée de probabilité "dans le pire cas" car deux hypothèses sous-jacentes sont que toute erreur de transmission est détectée sur le dernier bit de la trame et que, à chaque retransmission, le temps nécessaire pour gagner l' accès au bus est le plus grand possible.
Cette analyse a été utilisée dans [Rüdiger, 1998] où l' auteur introduit une fonction de coût C qui reflète la capacité du système à respecter ces échéances. L'objectif étant naturellement de minimiser l'espérance de cette fonction coût.

III.3] Temps d'atteinte des états bus-off et erreur-passive

Intérêt : Les mécanismes de confinement d' erreurs de CAN ont pour objectif de détecter des dysfonctionnements d' origine matériels et de neutraliser (état erreur passive) ou d' isoler les stations défectueuses (état bus-off). Néanmoins, il peut arriver qu' une simple succession d' erreurs de transmission entraîne le passage de la station dans un de ces deux modes de marche dégradés. Il s' agit donc d' évaluer la probabilité d' occurrence d' un tel événement pour, si cela se justifie, prévoir une stratégie de repli adéquate (exemple : sur déconnexion du contrôle-moteur, le correcteur de stabilité d'un véhicule utilisera l'information vitesse diffusée par les ABS).

Formalisme utilisé: probabilités (analyse Markovienne)

Références bibliographiques: [Navet and Song, 2001] pour a), [Gaujal and Navet, 2001] pour b) - codes sources Maple disponibles à l'adresse : http://www.loria.fr/~nnavet/code/fet01.html

III.3.a] Temps d'atteinte de l'état bus-off

Lorsque le compteur d'erreur en émission (TEC) atteint 256, toute station CAN se déconnecte du bus : c'est l'état bus-off. Nous pouvons estimer l'espérance du temps d'atteinte de l'état bus-off ainsi que la variance en faisant l'hypothèse que les changements d'états sont exponentiellement distribués. Sous cette hypothèse, l'évolution du TEC peut être modélisée par un processus de Markov.
La règle est que le TEC de la station émettrice est incrémenté de 8 si la trame émise est corrompue et décrémentée de 1 si la transmission est réussie néanmoins la valeur du TEC ne peut être négative et ne peut dépasser 256. Le taux de trames corrompues

de la station k est noté λ_1^k et le taux de trames transmises avec succès est λ_0^k. L' état 256, qui correspond au passage de la station dans l' état bus-off, est un état particulier, dit absorbant, duquel on ne peut ressortir une fois qu' il est atteint et qui stoppe le processus. Cela correspond exactement au comportement réel d' une station CAN. Avec les règles précédemment exposées, on obtient la matrice des probabilités de transition suivante de taille 257x257 notée Q :

	0	1	2	..	8	9	..	253	254	255	256
0	$-\lambda_1^k$	0	0	..	λ_1^k	0	..	0	0	0	0
1	λ_0^k	$-\lambda^k$	0	..	0	λ_1^k	..	0	0	0	0
2	0	λ_0^k	$-\lambda^k$	..	0	0	..	0	0	0	0
.	.	.	.	..	.	.	..	.	.	.	.
.	.	.	.	..	.	.	..	.	.	.	.
254	0	0	0	..	0	0	..	λ_0^k	$-\lambda^k$	0	λ_1^k
255	0	0	0	..	0	0	..	0	λ_0^k	$-\lambda^k$	λ_1^k
256	0	0	0	..	0	0	..	0	0	0	0

avec $\lambda^k = \lambda_0^k + \lambda_1^k$

Soit $q_i = \sum_{j \neq i} Q_{i,j}$ la somme des taux de sortie de l'état i et $q_{\max} = \sup_{i \geq 0} q_i$. Comme $q_{\max} < \infty$, ce processus de Markov (continu) peut être transformé en une chaîne de Markov en temps discret plus facile à manipuler. Cette chaîne, notée P, est définie par :

$$P_{i,j} = \begin{cases} Q_{i,j} / q_{\max}, & i \neq j \\ 1 - q_i / q_{\max}, & i = j \end{cases}$$

La forme canonique de la matrice P est : $P = \begin{bmatrix} Z & R \\ 0 & 1 \end{bmatrix}$ où Z est la matrice sans la 257$^{\text{ième}}$ ligne et 257$^{\text{ième}}$ colonne et R le 257$^{\text{ième}}$ vecteur colonne sans le 257 élément. On note T l'ensemble des états transients (probabilité non-nulle de non-retour, tous les états de Z) et N_i la v.a. qui donne le temps nécessaire pour atteindre pour la première fois l'état bus-off. Avec une analyse classique dite du « pas en avant », on obtient :

$$N_i = \begin{cases} \gamma_i + N_j, \text{avec proba} \sum_{j \in T} P_{i,j} \\ \gamma_i, \text{avec proba } P_{i,256} \end{cases}$$

avec $\gamma_i = 1$ si $i \neq 256$ et 0 sinon. En prenant les espérances, on obtient :

$$N_i = P_{i,256} E[\gamma_i] + \sum_{j \in T} P_{i,j} E[\gamma_i + N_j] = \gamma_i + \sum_{j \in T} P_{i,j} E[N_j]$$

Cet ensemble de 257 équations linéaires peut être résolu avec un logiciel de calcul numérique ou symbolique comme Maple. $E[N_0]$ est le temps moyen d'atteinte de bus-off pour la station considérée. Une analyse très similaire nous permet d'obtenir la variance des temps d'atteinte.

III.3.b] Temps d'atteinte de l'état erreur-passive

Une station erreur-passive n'est pas déconnectée du bus. Néanmoins elle doit attendre 8 bits supplémentaires après la fin d'une transmission pour pouvoir émettre une trame à son tour. Cela va nécessairement augmenter le temps de réponse des trames de la station et éventuellement les conduire à ne plus respecter les bornes calculées par analyse d'ordonnançabilité. Il est donc important d'évaluer la probabilité d'occurrence d'un tel événement.
Une station devient erreur-passive si le TEC est supérieur à 127 ou si le compteur d'erreurs en réception (REC) est égal à 128. Dans [Gaujal and Navet, 2001], nous avons montré que dans des conditions normales (la station a des trames à émettre), il était extrêmement improbable d'atteindre erreur-passive à cause de trames reçues (REC). Nous pouvons donc estimer le temps passé en erreur-passive comme le temps pendant lequel le TEC est dans un état > à 128. Soit M_i la v.a. qui donne le temps passé en erreur-passive avant que la station devienne bus-off. Son espérance est $M_i = \gamma_i + \sum_{j \in T} P_{i,j} E[M_j]$ avec $\gamma_i = 1$ si $i \geq 128$ et 0 sinon. Sur des applications numériques effectuées avec des modèles d'erreurs plausibles, nous remarquons que la proportion du temps passée en erreur-passive peut-être importante.

IV] La validation sur prototypes

Dans l' industrie, la construction de prototypes et de maquettes est souvent partie intégrante de la phase de conception. Par l' observation sur un prototype ou sur une maquette, il est possible, comme nous allons l' étudier sur un exemple, de vérifier le respect de certaines contraintes de niveau applicatif. D' autre part, l' observation nous permet de s' assurer de la pertinence de nos modèles, qu' ils soient analytiques ou de simulation, et de fixer au mieux les paramètres de ces modèles. Enfin un observateur réseau est indispensable pour s' assurer que des équipements fournis par des tiers respectent leur spécification.
Dans le cadre d' un contrat industriel avec PSA [Song and Navet, 1997], nous avons développé un observateur réseau "intelligent", appelé "Observer", qui enregistre toutes les trames échangées sur le réseau et vérifie (1) que le trafic a les caractéristiques voulues et (2) le respect de certaines contraintes temporelles préalablement spécifiées. Un langage de configuration a été conçu pour définir :

- les trames de l' application (identifieur, longueur des données, périodes),
- les informations (ou "signaux") à afficher (vitesse, couple),
- les contraintes de niveau applicatif à vérifier, sous la forme d'une contrainte de temps entre un stimulus et la réponse associée.

Le choix de ne pas se baser sur des outils du commerce (comme CAN-Analyzer de Vector) qui utilisent généralement des langages interprétés a été fait pour des questions de performances ; environ 700

trames sont à analyser par seconde dans l'application que nous avons étudiée.

Une contrainte typique dont le respect doit être vérifié dans une application véhicule est le délai entre un changement de vitesse avec une boite automatique (BVA) et la réduction de couple associé effectuée par le contrôle moteur (CM). Ce délai doit toujours rester inférieur à une certaine valeur, fixée par les concepteurs du véhicule en fonction de certains impératifs (mécaniques et de confort). Dans notre

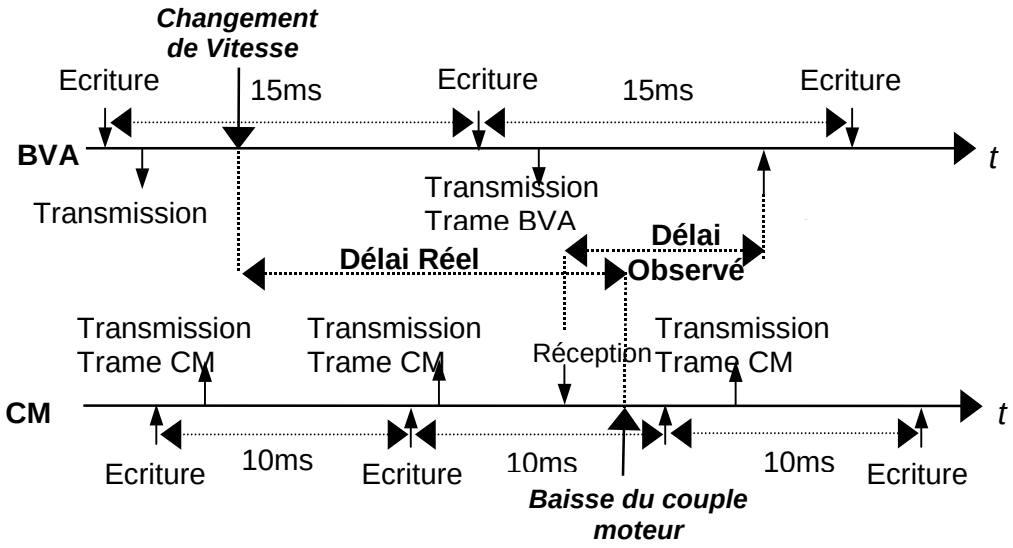

Figure 3 : délai stimulus-réponse – changement de vitesse – réduction de couple.

application, la BVA transmet toutes les 15 ms une trame contenant le rapport de vitesse engagé. De son coté, le contrôle moteur transmet le couple toutes les 10 ms. Lorsqu' un changement de vitesse est détecté par l' examen de la trame provenant de la BVA (le stimulus), Observer doit analyser chaque trame provenant du CM jusqu' à ce que le couple se situe en dessous de la valeur spécifiée (réponse au stimulus).

Comme cela est représenté sur la figure 3, nous remarquons que le délai que l' on peut observer (DO) entre le stimulus et la réponse associée, n' est qu' une estimation du délai réel (DR). Néanmoins, cette information est utile car elle nous permet de donner une borne sur le délai réel, en effet nous savons que

$$DR \leq T_{BVA} + J_{BVA} + DO + C_{BVA} - C_m$$

où T_{BVA} et J_{BVA} sont respectivement la période et la gigue de la trame contenant le rapport de vitesse, C_{BVA} étant son temps de transmission et C_{CM} le temps de transmission de la trame envoyée par le contrôle moteur.

V] Conclusions

Cet article se propose de donner une synthèse des technique de validation d' applications distribuées autour d' un réseau CAN. Le processus de validation est conduit en résolvant des modèles, analytiques ou de simulation, ainsi que par l' observation sur maquettes ou prototypes. Le tableau 2 résume les résultats que l' on peut attendre de chaque technique et leur complémentarité. Les techniques analytiques fournissent des bornes sur les métriques de performance considérées (temps de réponse, temps de réponse avec erreurs de transmission, probabilité de non-respect des échéances) ou permettent d' évaluer l' occurrence d' événements rares (temps d' atteinte de l' état busoff d' une station CAN). La simulation ne fournit pas de garanties mais des résultats plus proches du cas effectivement rencontré dans la pratique qui permettent au concepteur d' évaluer les performances moyennes de son application et de l' optimiser. Les modèles, qu' ils soient analytiques ou de simulation, mettent clairement l' accent sur le système de communication en utilisant des modèles très simples d' ECU. Il est naturellement possible de développer des modèles de simulation plus détaillés, comme cela a été entrepris dans [Courrier et al., 1998], néanmoins la validation des modèles, la fixation des paramètres et surtout l' interprétation des résultats deviennent alors problématiques.

D' autre part certaines stations du réseau fournies par des tiers existent généralement dès la phase de conception du système. Il est alors intéressant de les utiliser plutôt que des modèles. Ceci nous a conduit à développer un observateur réseau qui est bien adapté à la vérification de contraintes applicatif du type stimulus-réponse.

La validation d'une architecture construite en composant des mécanismes exécutifs existants (protocoles, ordonnanceurs, OS) passe par une étude des performances du système notamment par analyse d'ordonnançabilité ou avec une technique d'estimation probabiliste. Une stratégie complémentaire est de mettre au point des mécanismes exécutifs assurant par construction le respect des contraintes pendant l'exécution de l'application ou optimisant le système relativement à des métriques de performances choisies. Ces travaux font l'objet de recherches dans notre équipe avec par exemple la mise au point de politiques d'ordonnancement de tâches et de messages dans lesquels plusieurs types de trafic coexistent avec pour chacun des objectifs différents [Gaujal et al, 2001]. Il peut s'agir par exemple de minimiser le temps de réponse du trafic à contraintes souples tout en garantissant les temps de réponse du trafic à contraintes strictes.

Contraintes	Analyse	Simulation	Observation
Contraintes de temps • Bornes sur le temps de réponse o avec erreurs de trans. o avec prise en compte de l'OS support o de bout-en-bout • Temps de réponse moyen • Gigue en réception	 * * * *	 *	 *
Autres contraintes de SdF • Probabilité de non-respect des échéances • Temps moyen d'atteinte de l'état bus-off • Temps moyen d'atteinte de l'état error-passive	 * * *		
Contraintes de niveau applicatif • Délai stimulus-réponse			*

Tableau 2 : Les contraintes et leur(s) technique(s) de vérification

Références bibliographiques

nb : les papiers écrits par l'auteur sont téléchargeables à l'adresse http://www.loria.fr/~nnavet

[Simonot-Lion et al., 1995] F. Simonot-Lion, J.-P.Thomesse, M.Bayart, M.Staroswiecki, Dependable distributed computer control systems: analysis of the design step activities. In Proc. of the Conference IFAC-DCCS '95.

[Bélissent, 1996] P. Bélissent. Validation des systèmes temps réel distribués autour de CAN. Centre de Recherche en Informatique de Nancy (CRIN), Mémoire de DEA, UHP Nancy I, Septembre 1996.

[Castelpietra et al. 2001] P. Castelpietra, Y.Q. Song, F. Simonot-Lion, M. Attia, Carosse-Perf : A modular approach for simulation of in-vehicle embedded architectures. In 15th European Simulation Multiconference, ESM2001, Juin 2001.

[Courrier et al., 1998] M. Courrier, F. Simonot-Lion, and Y.Q. Song. Microscopic modeling of support system for in-vehicle embedded systems. In International IFIP Workshop on Distributed and Parallel Embedded Systems, DIPES' 98, 1998.

[Gaujal and Navet, 2001] B. Gaujal and N. Navet. CAN fault confinement mechanisms: analysis and improvements. Accepted for publication in Proc. FeT'2001, Nancy, Novembre 2001.

[Gaujal et al, 2001] B. Gaujal, N. Navet, J. Migge. "Dual-Priority versus Background Scheduling : a Path-wise Comparison", accepté en 2001 pour publication dans Real-Time Systems, Kluwer Academic Publishers. Version préliminaire disponible comme rapport de recherche INRIA RR-3734.

[Kiencke et al., 1998] U. Kiencke, J. Dirk, and S. Schneider. Performance analysis of a distributed automotive real-time system. In 4th international CAN Conference, ICC' 97, pages 7.02-7.08, 1997.

[Navet and Song, 1996] N. Navet and Y.-Q. Song. Evaluation de performances de la messagerie CAN du véhicule prototype PSA - action 1 du contrat PSA-CRIN. Centre de Recherche en Informatique de Nancy (CRIN), 1996. Rapport de fin de contrat 96-R-182.

[Navet and Song, 1998] N. Navet and Y.-Q. Song. On fault tolerance and worst-case response time analysis in CAN. In 23nd IFAC/IFIP Workshop on Real-Time Programming, WRTP' 98, Juin 1998.

[Navet and Song, 2001] N. Navet and Y.-Q. Song. Validation of in-vehicle real-time applications. Accepted for publication in Computers in Industry, to appear in 2001.

[Navet et al., 2000] N. Navet, Y.-Q. Song, and F. Simonot. Worst-case deadline failure probability in real-time applications distributed over CAN (Controller Area Network). Journal of Systems Architecture, 46 (7):607-618, 2000.

[Rüdiger, 1998] R. Rüdiger. Evaluating the tempora7l behaviour of CAN-based systems by means of a cost functional. In 5th international CAN Conference, ICC' 98, Novembre 1998.

[Song and Navet, 1997] Y.-Q. Song and N. Navet. Validation d' applications distribuées autour du réseau CAN par vérification en-ligne - développement d' un observateur réseau. Centre de Recherche en Informatique de Nancy (CRIN), 1997. Rapport de fin de contrat 97-R-110.

[Tindell and Burns, 1994] K. Tindell and A. Burns. Guaranteeing message latencies on Controller Area Network (CAN). In 1st International CAN Conference, ICC' 94, 1994.

[Tindell and Clark, 1994] K. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. Microprocessors and Microprogramming, 40:117-134, 1994.

[Tindell et al., 1995] K. Tindell, Burns, and A.J. Wellings. Calculating Controller Area Network (CAN) message response times. Control Engineering Practice, 3(8):1163-1169, 1995.

Short Biography :
Nicolas Navet received the BSc in Computer Science from the University of Mulhouse (France), the MSc in Computer Science from the Technische Fachhochschule Berlin (Germany) and the PhD from the Institut National Polytechnique de Lorraine (France) in 1999. He is now researcher at the INRIA (http://www.loria.fr/~nnavet). His research interests include validation of real-time applications and scheduling theory.